SYSTEM SIMULATION BY RECURSIVE FEEDBACK: COUPLING A SET OF STAND-ALONE SUBSYSTEM SIMULATIONS

Douglas D. Nixon
Vehicle and Systems Development Department
Marshall Space Flight Center
Huntsville, AL 35812 USA

Key Words: Systems Simulation, Systems Modeling, Numerical Coupling, Feedback

Abstract— **Recursive feedback is defined and discussed as a framework for development of specific algorithms and procedures that propagate the time-domain solution for a dynamical system simulation consisting of multiple numerically coupled self-contained stand-alone subsystem simulations. A satellite motion example containing three subsystems (orbit dynamics, attitude dynamics, and aerodynamics) has been defined and constructed using this approach. Conventional solution methods are used in the subsystem simulations. Centralized and distributed versions of coupling structure have been addressed. Numerical results are evaluated by direct comparison with a standard total-system simultaneous-solution approach.**

I. INTRODUCTION

Digital simulations of dynamical systems are often built by constructing algorithms that solve a set of differential-algebraic equations that mathematically model the system. The equations are solved numerically, as a single coupled set, using one of several standard or modified numerical integration methods. In some cases, software is written in a language such as FORTRAN or C to define the equations, and an existing or slightly modified ordinary differential equation (ODE) solver, perhaps a Runge-Kutta implementation, is employed to perform the integration. In other cases, an in-house development such as Marshall System for Aerospace Simulation (MARSYAS) or a commercial off-the-shelf product such as MATLAB®/SIMULINK® is used to act as a higher level facilitator of what is ultimately a mathematically similar approach.

System-level digital solution of coupled "stand-alone" dynamical simulations is a fundamentally different approach to system simulation, and fundamentally different numerical procedures are required. Recursive feedback is a conceptual method from which a family of appropriate algorithms, processes, and specific numerical procedures can be derived.

Coupled subsystem simulation architecture provides near complete independence of stand-alone sub-simulations, and naturally facilitates high fidelity and broad scope through collaboration across interfaces that can be implemented in the same physical and engineering terms that define them in the actual system. Such simplicity promotes clarity of communication and ease of understanding, both of which have many positive benefits. Individual sub-simulations can potentially be implemented on separate, remote, and dissimilar computational platforms, and this portends numerous advantages and possibilities as computer network capabilities improve.

II. RECURSIVE FEEDBACK ALGORITHM

A. *Concept*

Referring to Fig. 1, the subsystems are represented by independent, self-contained, time-domain, dynamical simulations that can be commanded to run from a given set of initial conditions over a predetermined segment of time when provided with input signals as functions of time over that interval. In general, the time segment, also referred to as a convergence interval, is short compared with the total duration of the simulation, but may be significantly larger than the integration step size associated with a typical subsystem. For each new stage of recursion, revised subsystem input signals are computed by summing system-level input signals with current stage feedback (coupling)

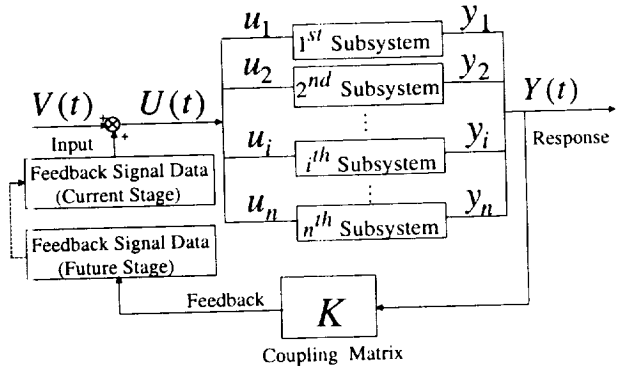

Figure 1: Schematic of Recursive Feedback Process

0-7803-7339-1/02/$17.00

signals that have been generated as if the loop were open. New subsystem responses and feedback signals are computed using the revised subsystem input signals in conjunction with original initial conditions. The process is started by ignoring the feedback signal, and continues recursively until convergence is achieved. After convergence, initial conditions are replaced with final conditions and the process is restarted for the next segment of time. System response over the total desired duration of the simulation is the concatenated set of combined subsystem responses over many time segments.

B. Insights for Specific Design

Because Fig. 1 describes a system of considerable generality whose complexity for a specific case can range from simple to great, further clarification of the process seems appropriate. The simplest "system" could arguably be a single integrator in the forward path, a constant multiplier (gain) in the backward (feedback) path, no (zero) system-level input, and a nonzero initial condition on the output. In that case, there is one linear subsystem, and that subsystem is a single integrator. The output function is the integral of the subsystem-level input function with respect to time, and can be generated independently and directly by simply computing area under the curve and adding the initial condition. The coupling matrix becomes a scalar (value of the loop gain), and actually represents direct feedback rather than coupling. The process, however, does not explicitly make such a distinction, nor does it need to. The summing junction becomes moot since there is no system-level input. The point at which feedback data is transferred from one stage to the next remains as depicted in Fig. 1, but does not exist in the analog system to be digitally simulated. The scalar differential equation that models the analog system is

$$\dot{y}(t) = ky(t) \quad , \tag{1}$$

where t is time and k is the loop gain, and the exact closed-form solution for the output response is given by the exponential function

$$y(t) = y(0)[e^{kt}] \quad , \tag{2}$$

where $y(0)$ represents the initial value of the output $y(t)$.

Applying recursive feedback to this system, the output is first computed as a function of time over the convergence interval as if there were no (zero) feedback. This function is referred to as the stage zero output response. Because the system input function is zero for this example, the response is simply a constant function of time symbolically represented by

$$y^{(0)}(t) = y(0) \quad , \tag{3}$$

where the parenthesized superscript is the stage index. Next, the stage zero feedback function is computed by multiplying the stage zero output response function by the loop gain k. After "moving" the feedback function numerical data across the recursive data transfer point (future stage to current stage in Fig. 1) and summing it with the system-level input function (zero), it becomes the stage one subsystem input function. Stage one response can now be computed by numerical integration of the new subsystem input function with application of the original initial condition $y(0)$. In the absence of numerical error, symbolic representation of the result is

$$y^{(1)}(t) = y(0)[1 + kt] \quad . \tag{4}$$

Repeating the process, symbolic representation of the stage two result is

$$y^{(2)}(t) = y(0)[1 + kt + \frac{(kt)^2}{2}] \quad , \tag{5}$$

By induction, the nth stage response is given by

$$y^{(n)}(t) = y(0)\left[1 + kt + \frac{(kt)^2}{2} + \cdots + \frac{(kt)^n}{n!}\right] \quad . \tag{6}$$

Clearly, if sufficiently continued, the symbolic recursive process yields a solution that approaches the exact solution (2). In theory, the time interval (convergence interval) for this case can be as long as desired; in practice, it must be restricted because of numerical integration and round-off errors.

As another example, consider the "system" to be an undamped oscillator governed by the scalar differential equation

$$\ddot{y}(t) + \omega^2 y(t) = 0 \quad , \tag{7}$$

where ω is the frequency of oscillation. A simulation of this system by recursive feedback can be set up in at least two very different ways. One design possibility is to define a single subsystem, best described as a double integrator, or two single integrators in series. The output function of the subsystem is determined by integrating its input function twice. The coupling matrix once again degenerates to a scalar constant, $-\omega^2$, and the system input function is zero. Assuming a nonzero initial condition $y(0)$ on the output (displacement), application of recursive feedback to this system results in a truncated series definition of $y(t)$, and at the nth recursion stage, a symbolic representation is

$$y^{(n)}(t) = y(0)\left[1 - \omega^2 \frac{t^2}{2} + \cdots (-1)^n \omega^{2n} \frac{t^{2n}}{(2n)!}\right] . \tag{8}$$

Then for a sufficiently large number of recursion stages, the approximate solution approaches the exact solution

$$y(t) = y(0)[\cos \omega t] \quad . \tag{9}$$

Once again the number of terms in the series corresponds directly to the number of recursion stages, but the order of

the series with respect to time is twice the number of recursion stages.

In contrast, a second design possibility is to define two subsystems that are both single integrators. The governing differential equation is now given by the two-dimensional matrix-vector equation

$$\dot{Y}(t) = KY(t) \ , \tag{10}$$

where the coupling matrix is given by

$$K = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} , \tag{11}$$

and the output function is given by

$$Y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} , \tag{12}$$

where $y_1(t)$ represents displacement and $y_2(t)$ represents velocity. Application of recursive feedback to this system results in a truncated series definition of $Y(t)$. At the nth recursion stage, the approximate system response function is symbolically represented by

$$Y^{(n)}(t) = \left[I + Kt + K^2 \frac{t^2}{2} + \cdots + K^n \frac{t^n}{n!} \right] Y(0) . \tag{13}$$

Again, each recursion stage, in effect, adds one term to the truncated series approximation of the exact solution.

For the two-subsystem, twin-integrator approach, it takes two recursion stages to reach the same order of approximation in time that was reached in one stage with the single-subsystem, double-integrator approach. However, the twin integrators represent independent operations that can be numerically performed in parallel; while the double integrator represents either a single operation, or two operations that must be performed in series. In the double integrator case, velocity, $\dot{y}(t)$, does not appear at the system level, only one function crosses the recursive data transfer point, and only one function is available to be checked for convergence at the system level. In the twin integrator case, velocity, $y_2(t)$, appears at the system level, $y_1(t)$ and $y_2(t)$ both cross the recursive transfer point, and both functions are available to be checked for convergence at the system level. Clearly, in a more complex situation, the relative merits of these approaches become simulation design and performance issues.

For the broader range of systems contained within the framework of Fig. 1, the nature of the coupling matrix is a function of many possible simulation design choices. In its most basic form, a coupler (element of the coupling matrix) would simply route signals from one subsystem to another without modifying them. In its most complex form, it could become a multidimensional nonlinear operator as well. As implied by the double integrator example when viewed as two single-integrator subsystems in series, it is also possible to distribute coupling functions throughout the system. Thus, the concept of a coupling matrix is not essential to the process; it may, however be useful in organization and control of a large simulation at the system level. In a particular situation, one might choose to view several subsystem simulations as a single combined set when the nature of the coupling among them is such that an output response can be determined directly from an input signal without necessity for recursion among members of the set. Typical coupler functions are likely to be coordinate transformations, gain multiplications, interpolations, and other data modification or routing tasks that must be performed to make proper interface connections among a set of predefined or existing sub-simulations.

C. Convergence

To be of practical value in simulation of dynamical systems, the ability of the recursion process to achieve convergence needs to be understood in a general sense so that reasonable assurance of convergence can be provided for specific circumstances. The convergence issue has been addressed for some simple nonlinear examples as well as a time varying example in [1]. For the nonlinear examples, each recursion stage adds higher order terms; but for a given number of correct terms in the series expansion, a greater number of recursion stages is required. For these systems, the convergence interval cannot be arbitrarily long, even without integration and round-off errors, because the series becomes divergent. The degree of restriction required for convergence depends on the initial condition.

The analysis of [2] addresses linearly coupled subsystems where the subsystems are also linear but are multi-dimensional. While subsystem output signals are linearly related to the subsystem states, the number of output signals may be less than the number of states. The recursive feedback process is analytically applied based only on inputs and outputs to the subsystems, as depicted in Fig. 1. Because no algebraic feed-through of subsystem input signals is allowed, the possibility of system-level algebraic loops is excluded. However, the analysis of [3], though much less straightforward, addresses a similar system with inclusion of system-level algebraic loops.

From this work and other experience, it appears that convergence can be achieved for a broad range of ordinary systems simply by controlling the length of the convergence interval. It is apparent, however, that this is

not always the case. For example, systems with lower-gain algebraic loops may converge, while those with higher gain may not. The essence of this problem can be understood by substituting a unit gain multiplication process for the integration process in the example system described by (1), and adding a non-zero input function. This creates a purely algebraic loop, and analytical application of the recursive feedback algorithm now generates a geometric series rather than an exponential series; and convergence becomes conditional. It should be noted that a non-convergent system of this type might also be physically untenable or not meaningful. The "guarantees" and insights with respect to convergence that come with analyses like that in [3] are weakened at best and totally lost at worst without the linear model assumption. However, one does have a guarantee of solution existence and uniqueness for a broad range of sufficiently well behaved nonlinear systems [4], and a converged recursive feedback process is indicative of a solution with exception of questions concerning discretization error, round-off error, inappropriate tolerances, etc. By shortening the length of the convergence interval, the number of recursion stages required to achieve convergence is normally decreased, or, in a non-convergent situation, the likelihood convergence is increased. The possibility exists for automatic in-process control of the convergence interval length as well as the number of recursion stages, not unlike control of stepsize and choice of order in a conventional algorithm for numerical solution of differential equations.

III. EXAMPLE SYSTEM

The example system consists of a satellite in low-Earth orbit that is experiencing aerodynamic effects in addition to gravitational effects. The spacecraft is idealized as a rigid body of rectangular "box" shape, illustrated in Fig. 2. The center of mass is offset from the geometric center by a small amount in all three axes, and moments of inertia are such that there are two large but somewhat unequal principal values, while the third principal value is much smaller. The principal axes are normal to the box surfaces so that all cross products of inertia are zero. The models are based on constant density atmosphere, "panel" aerodynamics and spherical Earth gravity. There is no attitude control, so the body is allowed to tumble under the influence of gravity and aerodynamics. Gravitational and aerodynamic torques affect the rotational dynamics of the rigid body, while gravitational and aerodynamic forces affect the translational dynamics. Both translational and rotational dynamics affect aerodynamic forces and torques, so a natural (analog) feedback loop is apparent. This example falls in the general category of nonlinear, continuous systems.

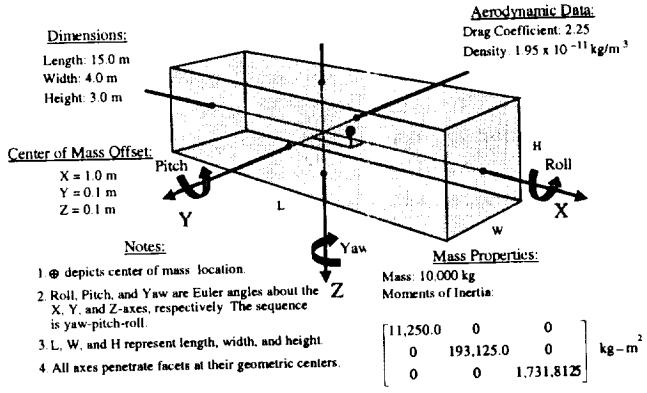

Figure 2: Spacecraft Properties and Coordinate System

IV. INDIVIDUAL SUBSYSTEM SIMULATIONS

A. Orbit Dynamics

The orbit dynamics simulation input signal is aerodynamic force, and output signals are orbit radius and velocity vectors. All orbit dynamics simulation variables are referenced to an Earth-centered inertial coordinate system as defined in Fig. 3. The underlying mathematical model, of the type found in [5], is represented by

$$m_{sat}\frac{d^2\vec{R}}{dt^2} = \vec{F}_{grav} + \vec{F}_{aero} \qquad (14)$$

and

$$\vec{F}_{grav} = -\frac{Gm_{Earth}m_{sat}}{R^3}\vec{R} \ , \qquad (15)$$

where G is the universal gravitational constant, m_{Earth} and m_{sat} are masses of the Earth and satellite, and $\vec{R}$ is the orbit radius vector. $\vec{R}$ extends from the center of the Earth to the center of mass of the satellite. $\vec{F}_{aero}$ is the aerodynamic force, and is derived from output of the aerodynamics subsystem simulation. After rearrangement to first-order form (six states), numerical solution of (14) is accomplished with a fourth-order fixed-step Runge-Kutta

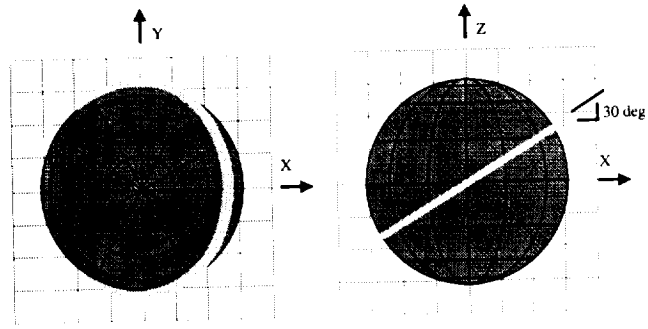

Figure 3: Inertial Coordinate System / Initial Orbit Plane Geometry

algorithm. Details of the formulation are given in [7].

B. Attitude Dynamics

Input signals for the attitude dynamics model are orbit radius vector expressed in inertial coordinates, and aerodynamic torque expressed in body coordinates. The body coordinate system is the same as the geometric coordinate system defined in Fig. 2 except that its origin is at the center of mass of the spacecraft. Output signals are body rates and attitude angles. The mathematical model, of the type found in [5], represents rotational motion of a rigid body, and is defined by

$$I \cdot \dot{\bar{\omega}} + \bar{\omega} \times I \cdot \bar{\omega} = \vec{T}_{gg} + \vec{T}_{aero} \quad , \qquad (16)$$

where I is the moment of inertia matrix, and $\bar{\omega}$ is the angular velocity vector associated with the body frame. $\vec{T}_{gg}$, the gravity gradient torque, is given by [6]

$$\vec{T}_{gg} = \frac{Gm_{Earth}}{R^3} \hat{r} \times I \cdot \hat{r} \quad , \qquad (17)$$

where $\hat{r}$ is a unit vector corresponding to $\vec{R}$ expressed in body coordinates. $\vec{T}_{aero}$, the aerodynamic torque, is derived from output of the aerodynamics subsystem simulation. Equation (16) is solved using an Euler angle formulation of the kinematics followed by rearrangement to first-order form (six states) and numerical integration by Runge-Kutta. Attitude angles of z-y-x Euler rotation sequence define body frame orientation with respect to the inertial frame. Details of the formulation are given in [7].

C. Aerodynamics

The aerodynamics simulation input signals are orbit radius vector in inertial coordinates and orbit velocity vector in body coordinates. Output signals are aerodynamic force and torque expressed in geometric coordinates. The spacecraft is modeled as a box with six rectangular side panels (Fig. 2). Drag force for the ith panel is defined empirically by [6]

$$\vec{F}_i = \begin{cases} -\frac{1}{2}\rho V^2 C_d A_i (\hat{n}_i \cdot \hat{v}) \hat{v} & \text{if} \quad \hat{n} \cdot \hat{v} \geq 0 \\ 0 & \text{if} \quad \hat{n} \cdot \hat{v} = 0 \end{cases} , \qquad (18)$$

where ρ is atmospheric density, V is velocity magnitude relative to the atmosphere, C_d is a drag coefficient, A is panel area, $\hat{n}$ is a unit vector normal to the panel and directed outward from the box, and $\hat{v}$ is a unit vector corresponding to $\vec{V}$ expressed in geometric coordinates. Because rotation of the Earth is neglected, no distinction between relative and absolute velocity is made. The total aerodynamic force is given by

$$\vec{F} = \sum_{i=1}^{6} \vec{F}_i \quad . \qquad (20)$$

The aerodynamic torque about the geometric center of the box is given by

$$\vec{T} = \sum_{i=1}^{6} \vec{r}_{cpi} \times \vec{F}_i \quad , \qquad (19)$$

where $\vec{r}_{cpi}$ is a vector directed from the origin of the geometric coordinate system to the center of pressure of the ith panel. Finally, $\vec{T}_{aero}$, as required in (16), is given by

$$\vec{T}_{aero} = \vec{T} - (\vec{r}_{cm} \times \vec{F}) \quad , \qquad (20)$$

where $\vec{r}_{cm}$ is a vector directed from the geometric center of the spacecraft to its center of mass. Equation (20) is not contained within the aerodynamics subsystem simulation, but is implemented as a coupler function.

V. SYSTEM SIMULATION STRUCTURES

Three system-level simulations of different structure have been built. The first, illustrated in Fig.4, uses an implementation of recursive feedback with centralized coupling, while a second, illustrated in Fig. 5, uses distributed coupling. The third uses a conventional first-order, single-equation-set approach and is the standard against which the recursive feedback approaches are compared. For the distributively coupled simulation, the subsystems have been arranged so that, to the extent possible, feedback is minimized. Because there is an information loop that ultimately must be closed, it is not possible to completely eliminate feedback. The sub-simulations are identical to those of the centrally coupled system, but the system-level numerical process is

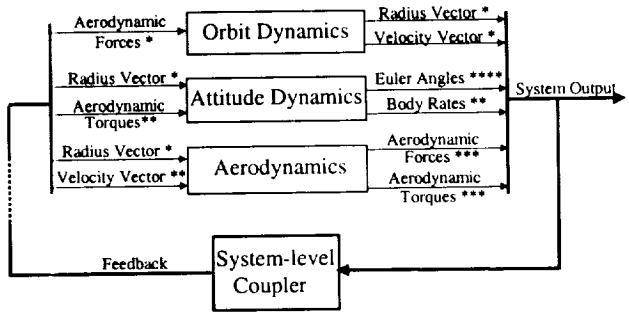

Figure 4: Centrally Coupled Simulation Diagram

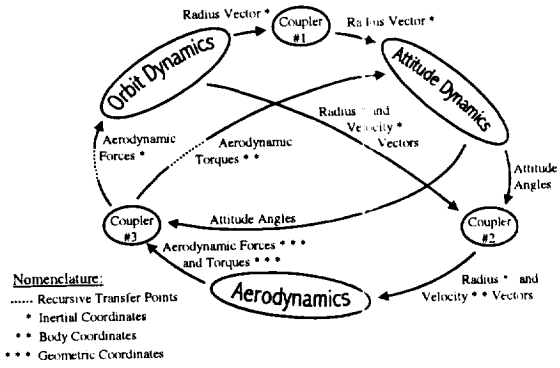

Figure 6: Distributively Coupled Simulation Diagram

substantially affected because, among other things, the number of scalar signal paths that cross the recursive transfer point has changed.

Simulations for each of the three subsystems have been constructed as functionally separate entities and combined within the framework of a single FORTRAN computer program that couples them by way of a recursive feedback process. Input and output signals (functions of time) are represented by a sequence of linearly connected points equally spaced in time, and additional points are obtained through interpolation by each simulation if needed The orbit dynamics and attitude dynamics simulations have separate numerical integration processes, while the aerodynamics simulation has no integration process because the model is purely logical-algebraic. All three simulations are "stand-alone" because each, in principle, is capable of producing output signals from input signals and initial conditions. Initial conditions, of course, do not apply to the aerodynamic simulation because no integration is involved.

The conventional simulation is implemented from the combined set of equations that define the subsystem models. They have been collected and rearranged, by hand, to a single set of 12 first-order differential equations, an arrangement sometimes called a "state variables" approach to system formulation. In that context, the state vector is defined as

$$X = \left[R \mid V \mid \Phi \mid \Omega\right]^T , \qquad (21)$$

where R, V, Φ and Ω are three-element row vectors representing orbit radius, velocity, attitude angles, and body rates. The system differential equations can now be represented as a first-order set by

$$\dot{X} = f(X) . \qquad (22)$$

A fourth-order fixed-step Runge-Kutta numerical integration algorithm specifically designed to solve systems in the form of equation (13) is employed to propagate the system responses.

IV. NUMERICAL RESULTS

A. Definition of the Run Case

Spacecraft mass properties, dimensions, and aerodynamic data are specified in Fig 2. Initial attitude angles and body rates are zero. The orbit is initialized so that in the absence of aerodynamic drag it would be circular at an altitude of 300 km and an inclination of 30 deg. The orbit initialization point is in the X–Z plane of the coordinate system shown in Fig. 3. The spacecraft is allowed to tumble under the influence of gravity and aerodynamics for a time period of 60,000 sec.

Algorithm parameters in both recursive method simulations were set to use five sample points per convergence interval (including both end points), and the interval length was 0.4 sec. RMS normalized and absolute convergence error tolerances were 10^{-12} for the centrally coupled case and 10^{-14} for the distributively coupled case. The integration step size for the orbit and attitude dynamics sub-simulations was 0.1 sec., and the aerodynamics sub-simulation computed aerodynamic effects at a 0.1 sec sample interval. Integration step size for the conventional simulation was also 0.1 sec.

B. Comparison of Responses

Responses from all the simulations agree for approximately 45,000 sec. of the 60,000 sec. duration; after that, the recursive method responses begin to diverge from the conventional method. However, the recursive methods continue to agree with each other for the full 60,000 sec. Figs. 6-12 present plots of time segments specifically selected to illustrate the early agreement and subsequent deviation of responses. Attitude angles are shown in Fig. 6–8, body rates are shown in Fig. 9-11, and altitude is shown for the final 1000 sec. in Fig. 12. Deviations are attributed primarily to the fact that aerodynamic forces and torques are linearly interpolated over the sub-simulation integration step size in the recursive methods, while they are computed at points internal to the integration interval in the conventional implementation. Improvement could likely be found through a more sophisticated interpolation method, a shorter convergence interval, or an increased number of points per convergence interval for signal definition. It must be remembered, however, that eventual disagreement of dynamical simulations driven by different numerical processes is always expected. Additional cases involving two- and three-axis attitude control were studied in [7], and no significant deviation became apparent for the full 60,000 sec. duration.

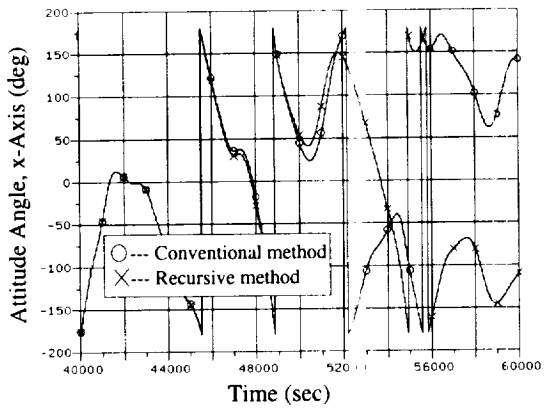

Figure 6: x-Axis Attitude Angle vs. Time

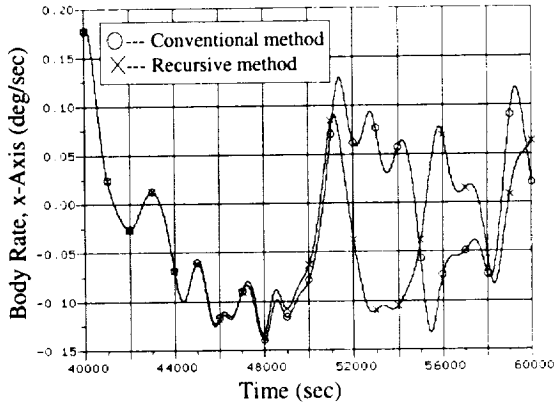

Figure 9: x-Axis Body Rate vs. Time

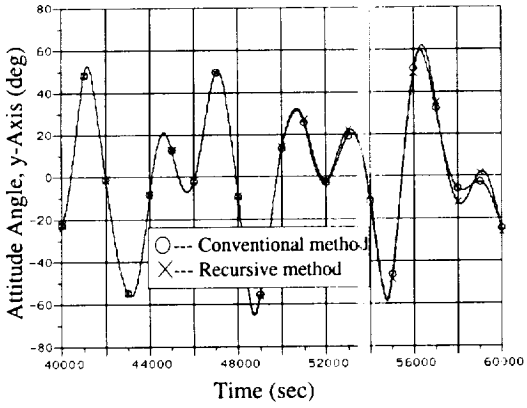

Figure 7: y-Axis Attitude Angle vs. Time

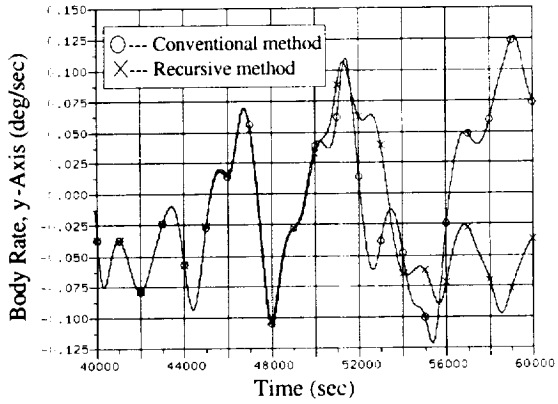

Figure 10: y-Axis Body Rate vs. Time

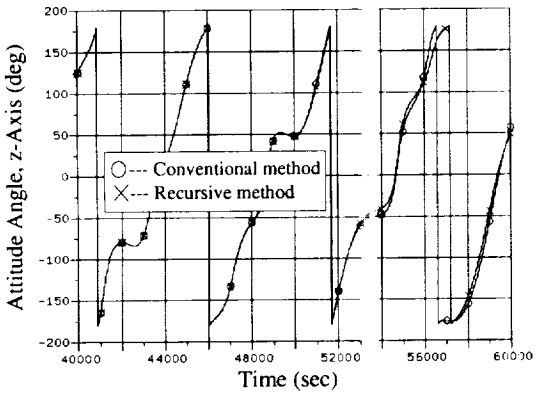

Figure 8: z-Axis Attitude Angle vs. Time

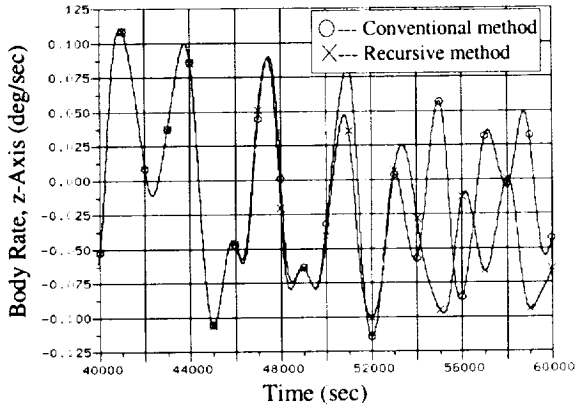

Figure 11: z-Axis Body Rate vs. Time

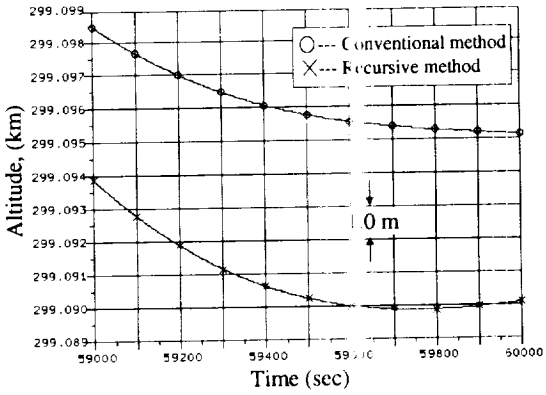

Figure 12: Altitude vs. Time

Performance.

For the example presented, the recursive methods were slower than the conventional method by a factor of 4 to 6 depending mainly on the coupling approach. The centrally coupled version was slowest, but no advantage of the possibility for parallel execution of subsystem simulations was taken in the current single-computer implementation. A multi-platform (or multi-processor) implementation could naturally take such advantage; however, that possibility does not exist in the distributively coupled version. In the recursive methods, response at the latest stage is compared with response from the previous stage to determine convergence with respect to a set of error tolerances. In the conventional method, no active control of error is present, and inclusion of such a mechanism would require additional computation. Conversely, predetermination of the number of recursion stages and elimination of the convergence check could significantly reduce the computation load for the recursive methods. While it is recognized that speed is important, none of the simulations were refined for efficiency, and it believed that many advantages of the recursive feedback approach lie elsewhere.

VI. SUMMARY AND CONCLUSIONS

An example nonlinear, continuous simulation of satellite motion has been successfully constructed using two variations of recursive feedback to couple three separate sub-simulations of orbit dynamics, attitude dynamics, and aerodynamics. Results have been verified by direct comparison with a conventionally constructed simulation of the same system. Each sub-simulation deals with one engineering discipline, and appropriate interactions are implemented recursively at the system level. The sub-simulations can, in principle, be run separately on remote and dissimilar platforms while coupling is facilitated by way of network. The system simulation can be designed so that coupling signals represent physically identifiable variables associated with physical interfaces among the subsystems. The example employs three sub-simulations; the method framework accommodates an arbitrary number. Clearly, further investigation of recursively coupled sub-simulations as a multidiscipline system simulation architecture is warranted.

REFERENCES

[1] ED11 (12-98-251), Memorandum for the Record, "An Assorted Collection of Analyses that Support the Concept of Recursive Feedback for Simulation of Dynamic Systems", D. Nixon, MSFC, December 1998.

[2] ED11 (12-98-255), Memorandum for the Record, "Analytical Demonstration of Solution by Recursive Feedback for a Linear System Composed of Internally Inaccessible Subsystems", D. Nixon, MSFC, December 1998.

[3] TD55 (00-18), Memorandum for the Record, "Analytical Demonstration of Recursive Feedback as a Solution Method for Simulation of Linear Dynamical Systems with Algebraic Loops", D. Nixon, MSFC, August 2000.

[4] Ogata, K.: *State Space Analysis of Control Systems*, Printice-Hall, Englewood Cliffs, NJ, pp. 12-15, 1967.

[5] Greenwood, D. T.: *Principles of Dynamics*, Printice-Hall, Englewood Cliffs, NJ, 1965.

[6] Wertz, J.R., and Wiley, J.L.: *Spacecraft Mission Analysis and Design*, 3rd ed., Microcosm Press Torrance, CA, p. 324, 1999.

[7] Nixon, D.D., "System Simulation by Recursive Feedback: Coupling a Set of Stand-Alone Subsystem Simulations", NASA/TP-2001-211331, October 2001.